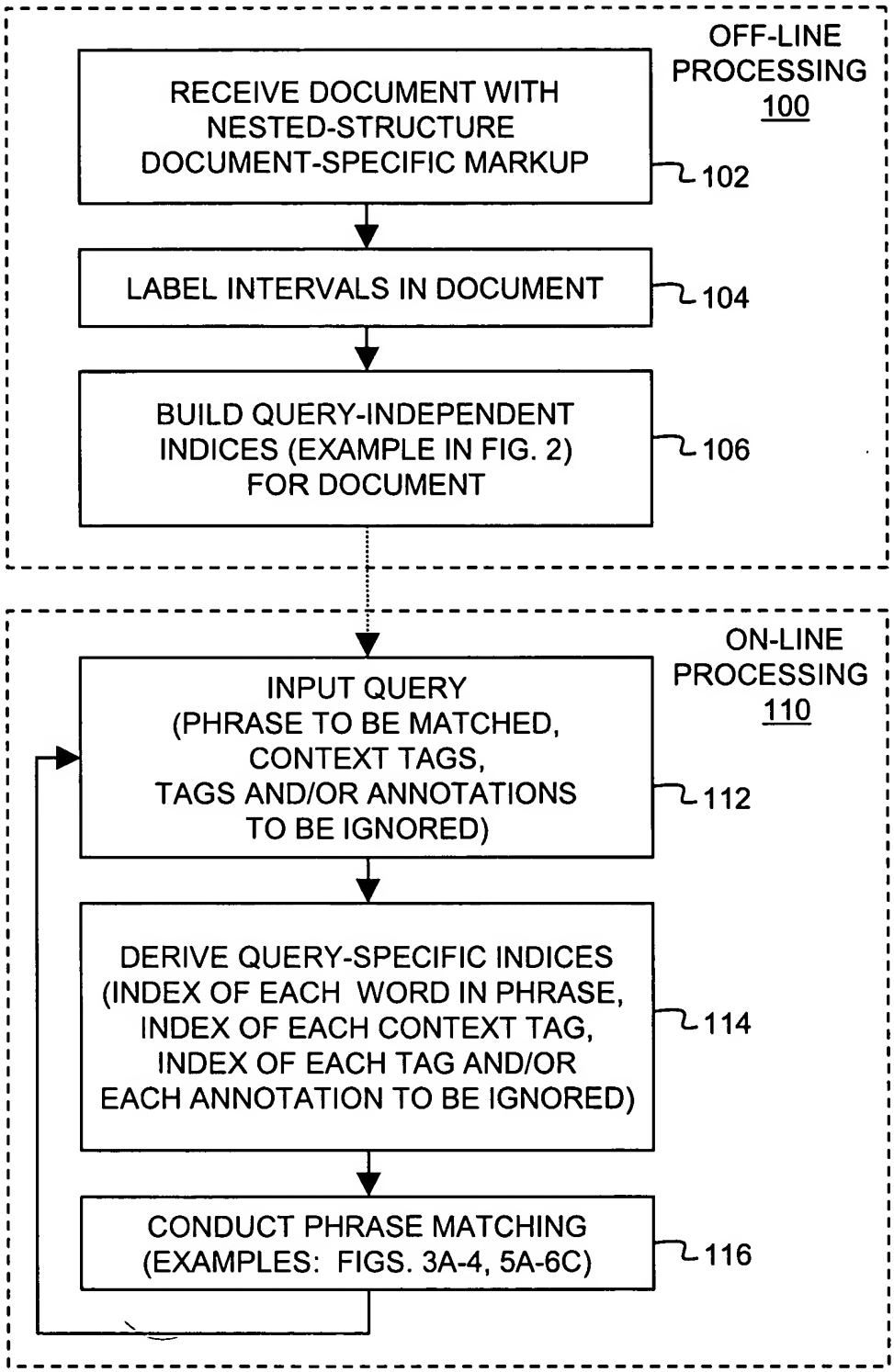
OFF-LINE PROCESSING 100
RECEIVE DOCUMENT WITH NESTED-STRUCTURE DOCUMENT-SPECIFIC MARKUP
102
LABEL INTERVALS IN DOCUMENT
104
BUILD QUERY-INDEPENDENT INDICES (EXAMPLE IN FIG. 2) FOR DOCUMENT
106
ON-LINE PROCESSING 110
INPUT QUERY (PHRASE TO BE MATCHED, CONTEXT TAGS, TAGS AND/OR ANNOTATIONS TO BE IGNORED)
112
DERIVE QUERY-SPECIFIC INDICES (INDEX OF EACH WORD IN PHRASE, INDEX OF EACH CONTEXT TAG, INDEX OF EACH TAG AND/OR EACH ANNOTATION TO BE IGNORED)
114
CONDUCT PHRASE MATCHING (EXAMPLES: FIGS. 3A-4, 5A-6C)
116

FIG. 1

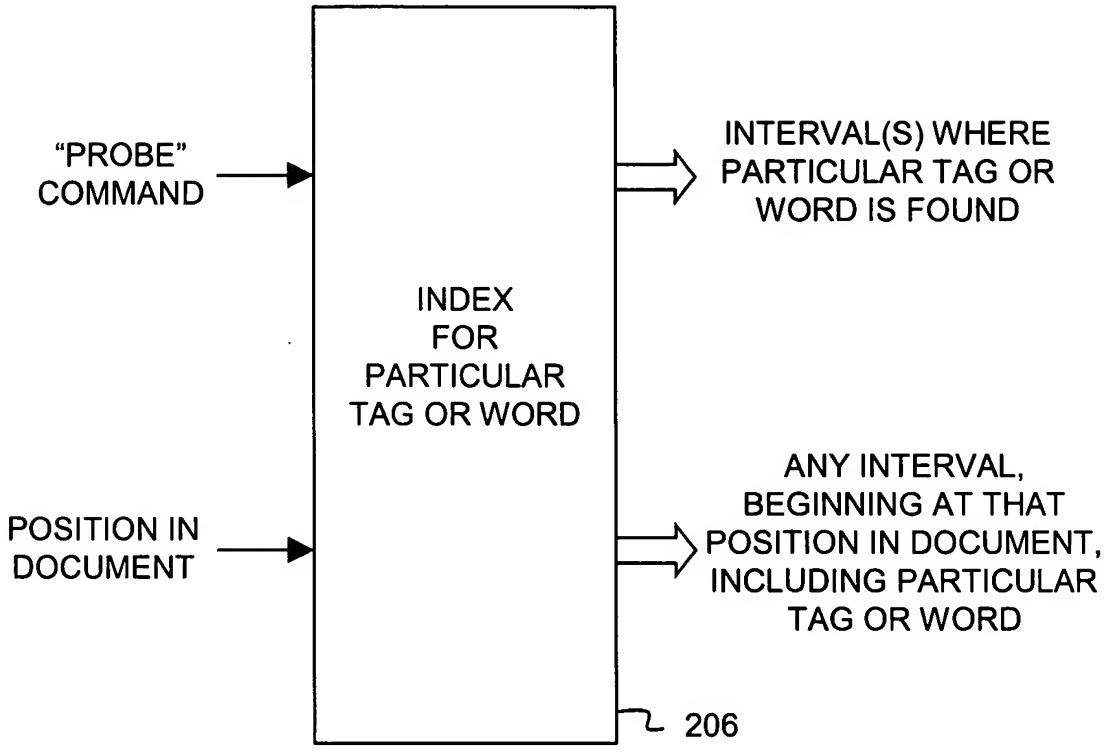
"PROBE" COMMAND
POSITION IN DOCUMENT
INDEX FOR PARTICULAR TAG OR WORD
INTERVAL(S) WHERE PARTICULAR TAG OR WORD IS FOUND
ANY INTERVAL, BEGINNING AT THAT POSITION IN DOCUMENT, INCLUDING PARTICULAR TAG OR WORD
206

FIG. 2

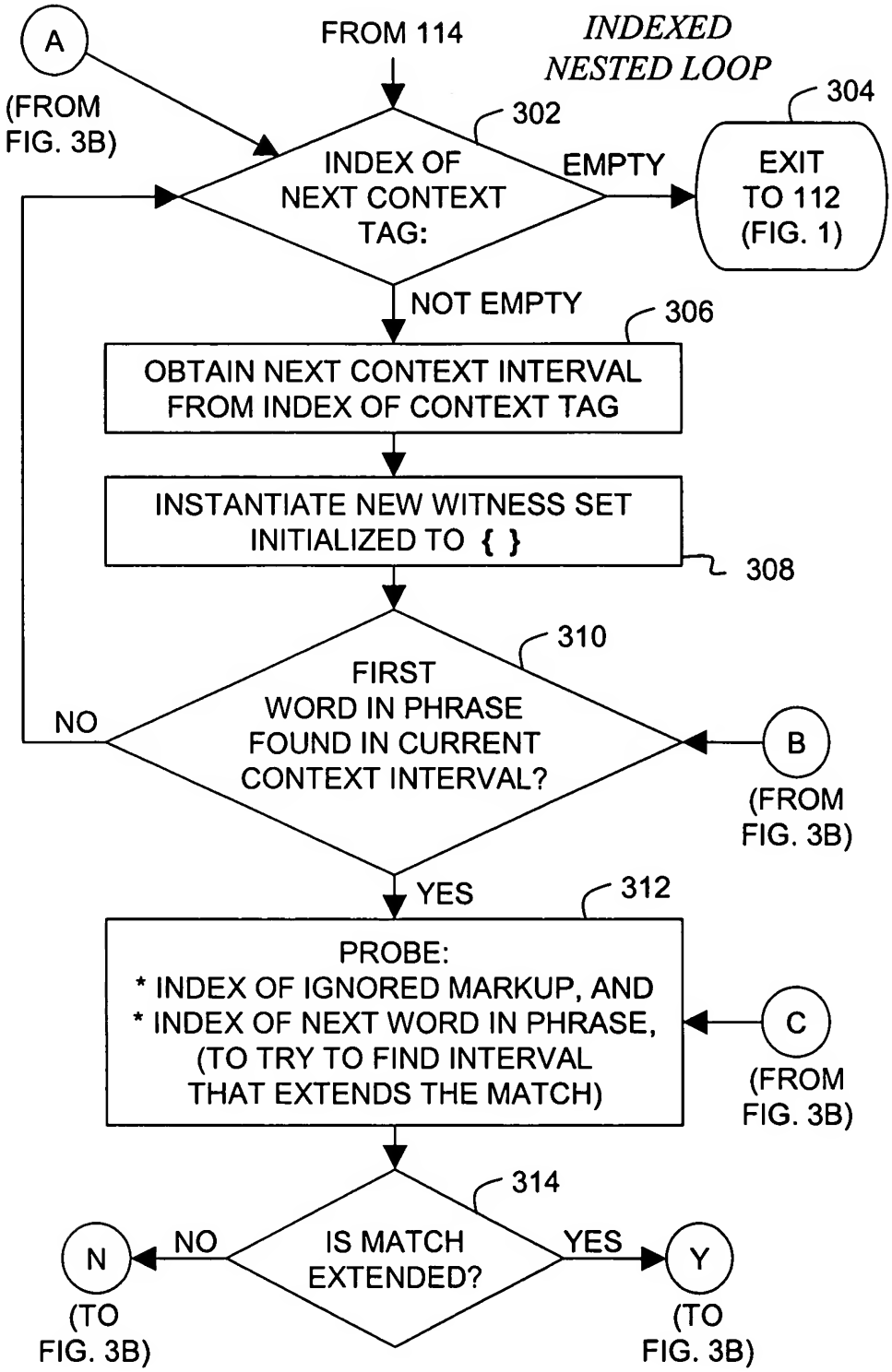
A
(FROM FIG. 3B)
FROM 114
INDEXED NESTED LOOP
302
INDEX OF NEXT CONTEXT TAG:
EMPTY
304
EXIT TO 112 (FIG. 1)
NOT EMPTY
306
OBTAIN NEXT CONTEXT INTERVAL FROM INDEX OF CONTEXT TAG
INSTANTIATE NEW WITNESS SET INITIALIZED TO { }
308
310
FIRST WORD IN PHRASE FOUND IN CURRENT CONTEXT INTERVAL?
NO
B
(FROM FIG. 3B)
YES
312
PROBE:
* INDEX OF IGNORED MARKUP, AND
* INDEX OF NEXT WORD IN PHRASE,
(TO TRY TO FIND INTERVAL THAT EXTENDS THE MATCH)
C
(FROM FIG. 3B)
314
IS MATCH EXTENDED?
NO
N
(TO FIG. 3B)
YES
Y
(TO FIG. 3B)

FIG. 3A

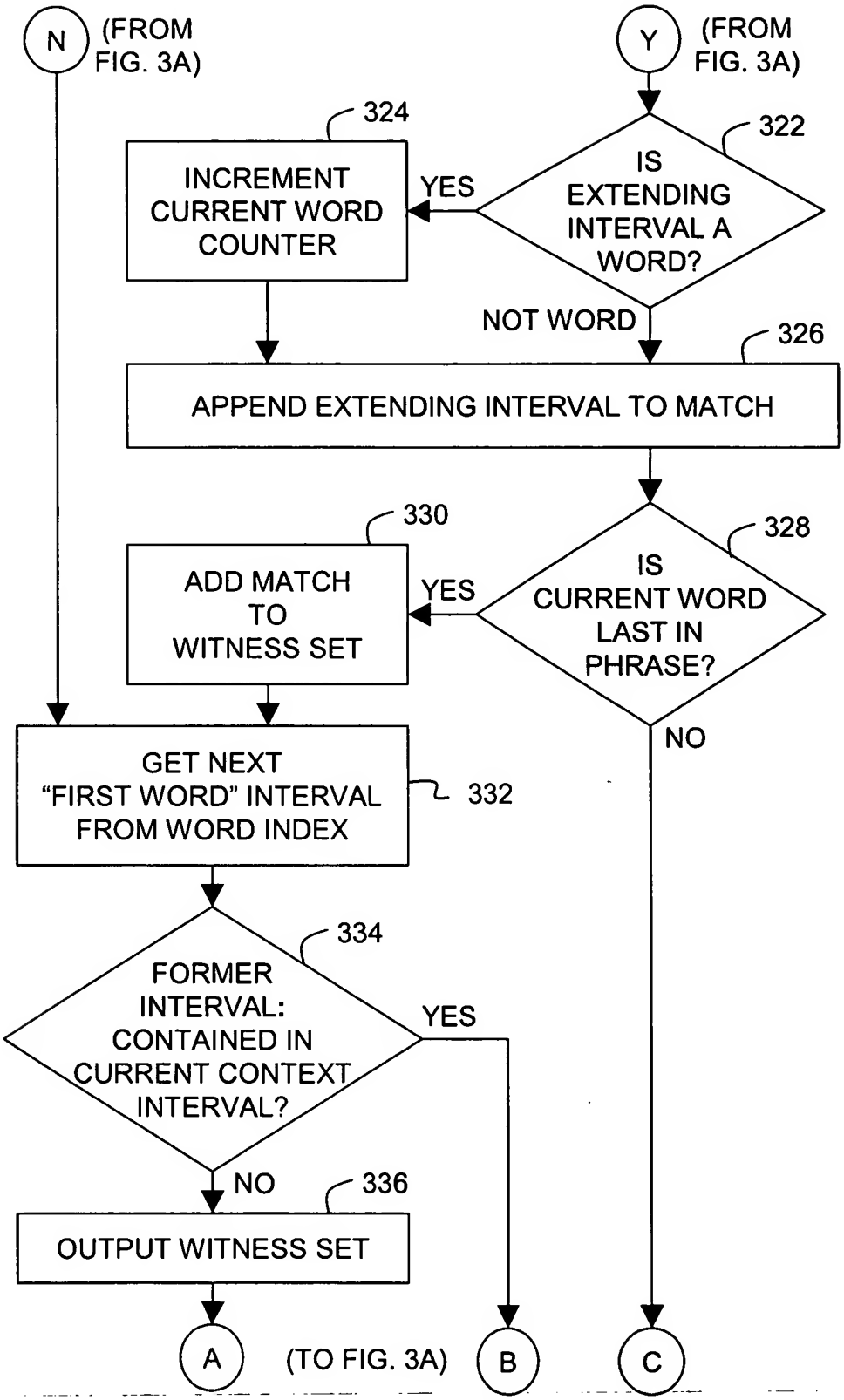
N
(FROM FIG. 3A)
Y
(FROM FIG. 3A)
324
INCREMENT CURRENT WORD COUNTER
YES
322
IS EXTENDING INTERVAL A WORD?
NOT WORD
326
APPEND EXTENDING INTERVAL TO MATCH
330
ADD MATCH TO WITNESS SET
YES
328
IS CURRENT WORD LAST IN PHRASE?
NO
GET NEXT "FIRST WORD" INTERVAL FROM WORD INDEX
332
334
FORMER INTERVAL: CONTAINED IN CURRENT CONTEXT INTERVAL?
YES
NO
336
OUTPUT WITNESS SET
A
(TO FIG. 3A)
B
C

FIG. 3B

```
0.   /* Indexed Nested Loop (INL) Approach  */

1.   for each context interval ic in LC {
2.     witnessSet = { }
3.     index probe Lw1 to find first interval i1
4.        such that descendant(i1, ic)
5.     repeat {
6.       matchPos = 1;
7.       m = [ i1 ];
8.       repeat {
9.         probe (Lw(matchP os+1) U LM) to find
10.          i2 with i2.start = last(m).end+1;
11.        if (no match found) break;
12.        if (i2 € Lw(matchPos+1)) matchPos++;
13.        m = append(m, i2);
14.        /* Matched last word in phrase */
15.      } until (matchPos = q)
16.      /* If a complete witness is found, save it */
17.      if (matchPos = q)
18.        witnessSet = witnessSet U { m };
19.      i1 = next(Lw1)
20.    } until not(descendant(i1, ic))
21.    output (ic, witnessSet)
22.  }

23.  descendant(i1, i2) {
24.    i1.start > i2.start and i1.end < i2.end
25.  }
```

FIG. 4

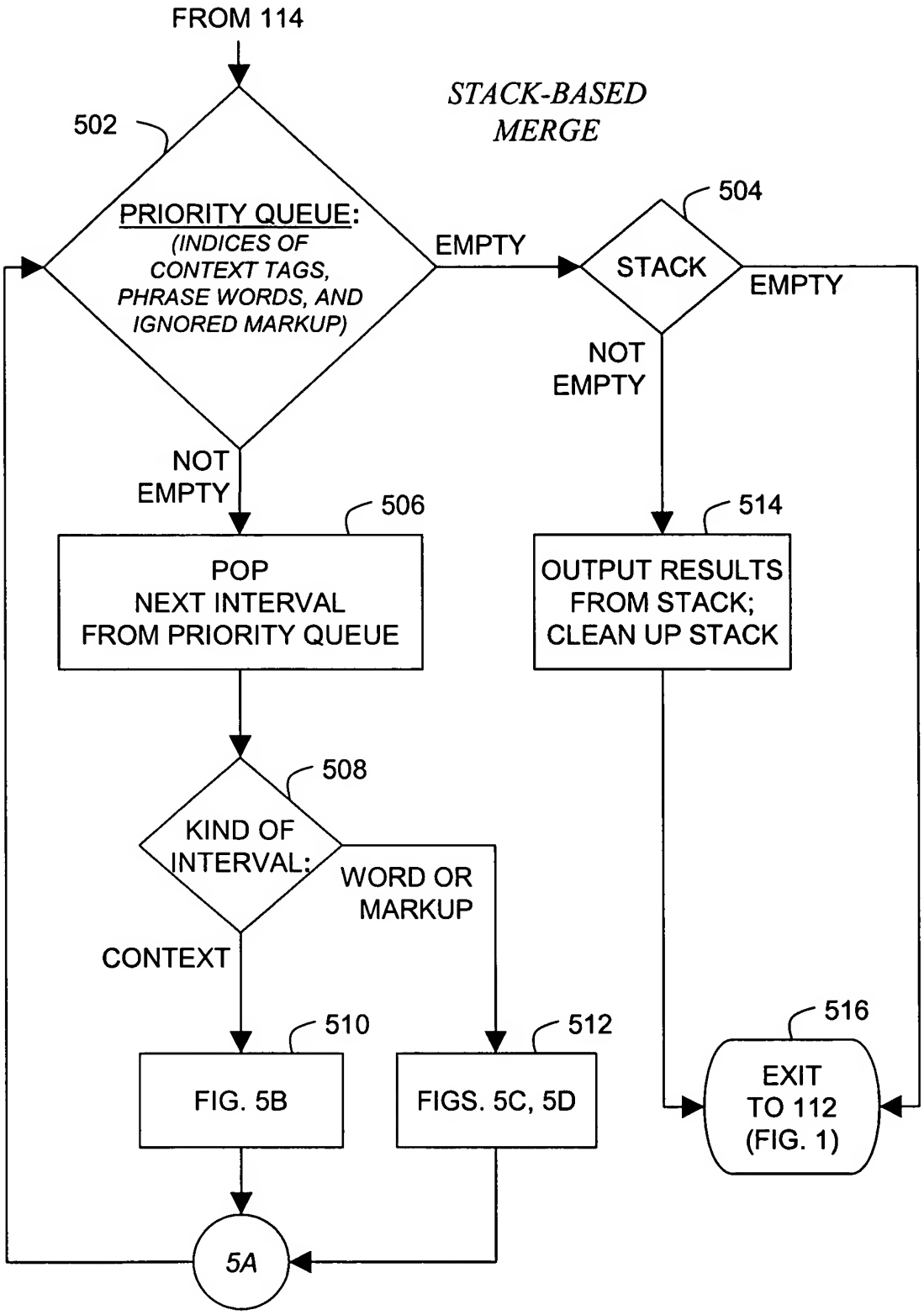
FROM 114
STACK-BASED MERGE
502
PRIORITY QUEUE: (INDICES OF CONTEXT TAGS, PHRASE WORDS, AND IGNORED MARKUP)
EMPTY
504
STACK
EMPTY
NOT EMPTY
NOT EMPTY
506
POP NEXT INTERVAL FROM PRIORITY QUEUE
514
OUTPUT RESULTS FROM STACK; CLEAN UP STACK
508
KIND OF INTERVAL:
WORD OR MARKUP
CONTEXT
510
FIG. 5B
512
FIGS. 5C, 5D
516
EXIT TO 112 (FIG. 1)
5A

FIG. 5A

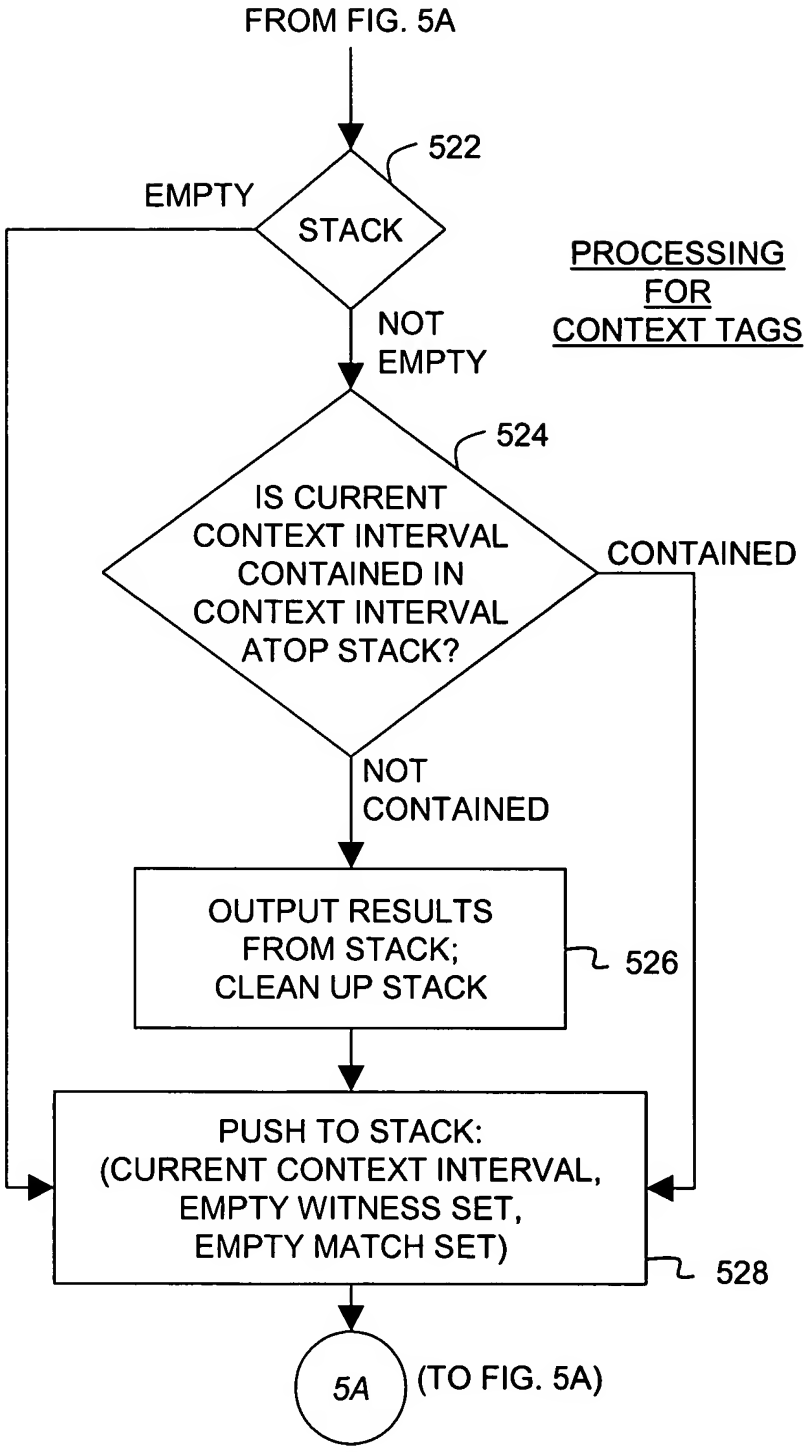
FROM FIG. 5A
522
EMPTY
STACK
PROCESSING FOR CONTEXT TAGS
NOT EMPTY
524
IS CURRENT CONTEXT INTERVAL CONTAINED IN CONTEXT INTERVAL ATOP STACK?
CONTAINED
NOT CONTAINED
OUTPUT RESULTS FROM STACK; CLEAN UP STACK
526
PUSH TO STACK: (CURRENT CONTEXT INTERVAL, EMPTY WITNESS SET, EMPTY MATCH SET)
528
5A
(TO FIG. 5A)

FIG. 5B

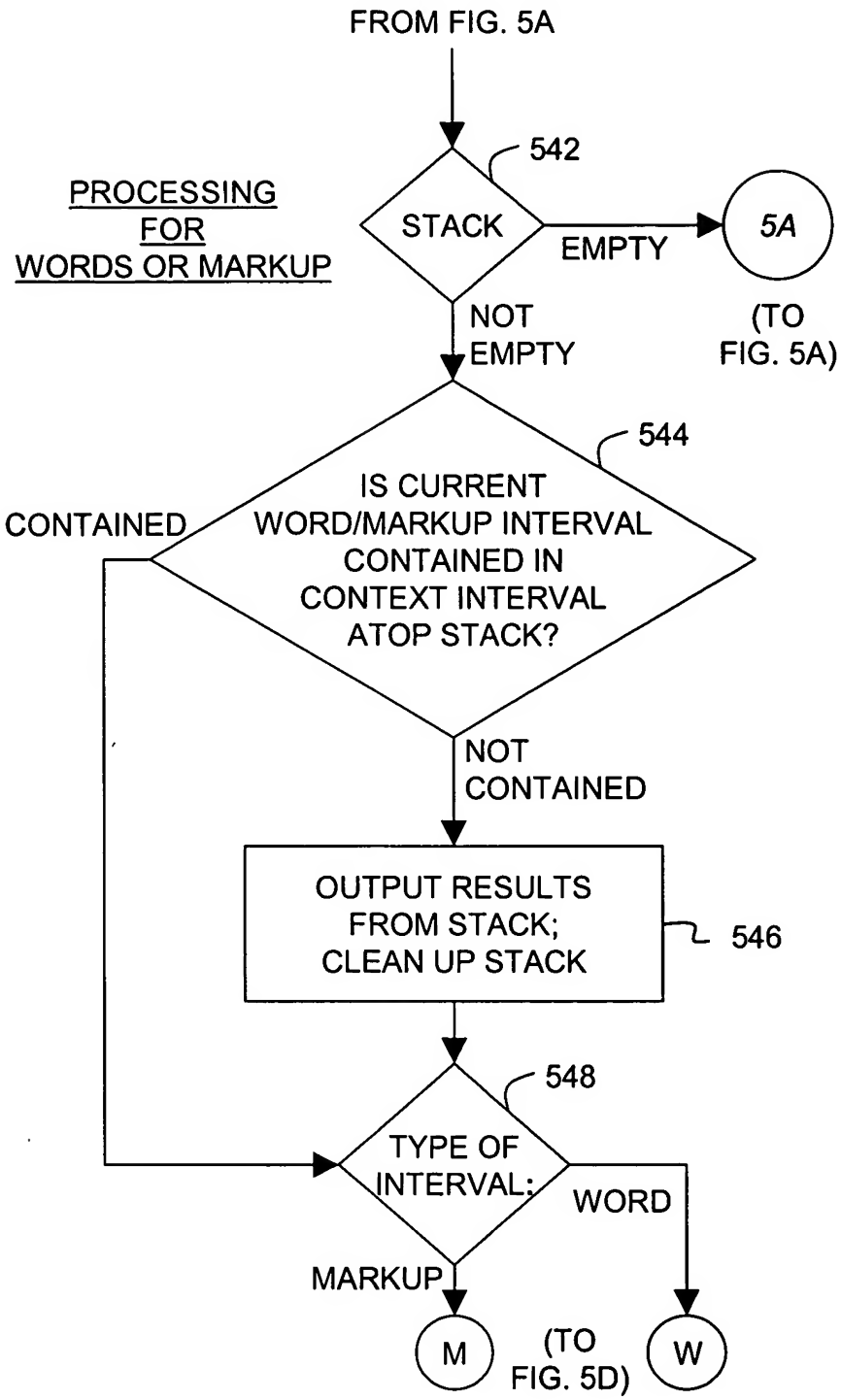
FROM FIG. 5A
PROCESSING FOR WORDS OR MARKUP
542
STACK
EMPTY
5A
(TO FIG. 5A)
NOT EMPTY
544
IS CURRENT WORD/MARKUP INTERVAL CONTAINED IN CONTEXT INTERVAL ATOP STACK?
CONTAINED
NOT CONTAINED
OUTPUT RESULTS FROM STACK; CLEAN UP STACK
546
548
TYPE OF INTERVAL:
WORD
MARKUP
M
(TO FIG. 5D)
W

FIG. 5C

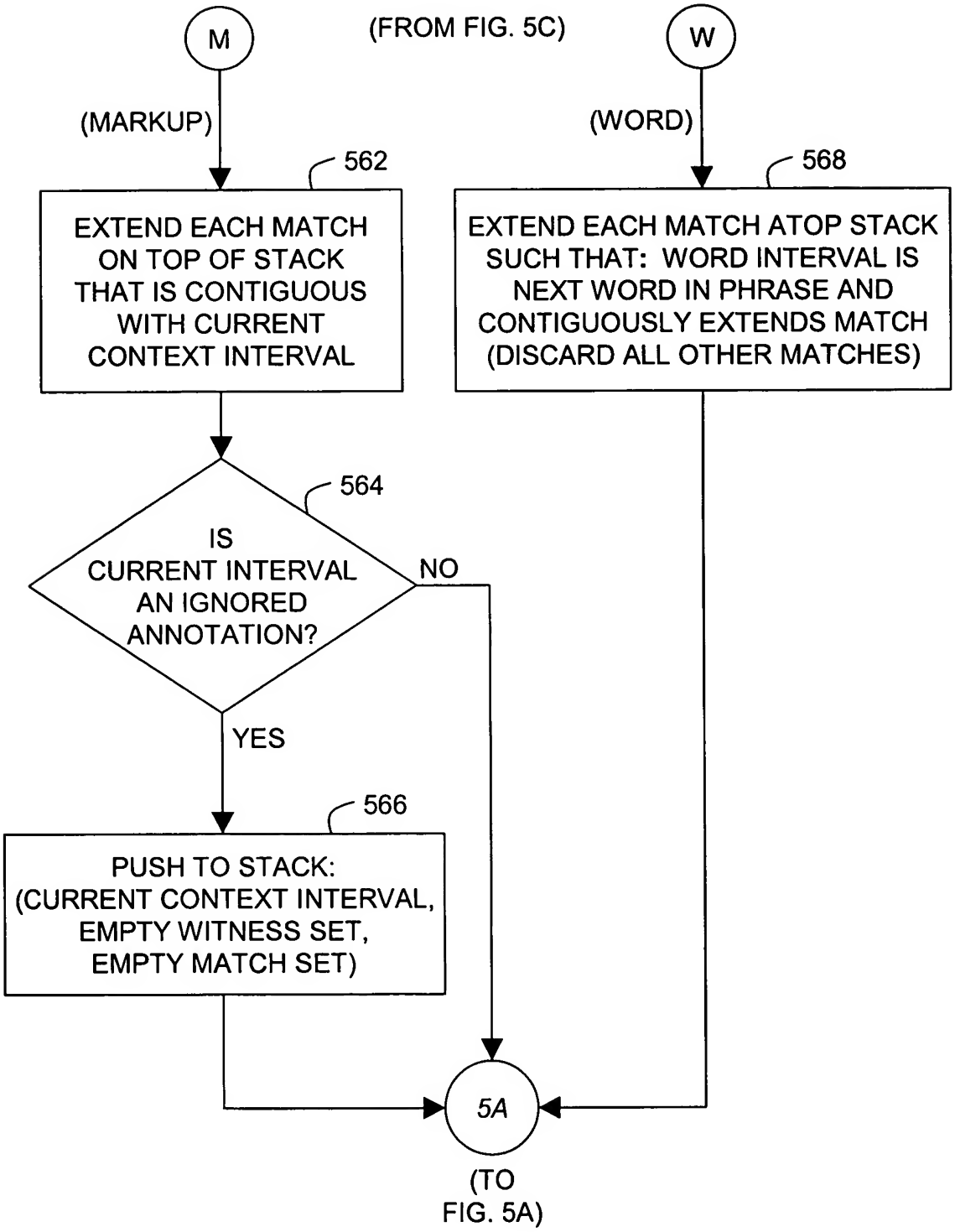
M
(FROM FIG. 5C)
W
(MARKUP)
(WORD)
562
568
EXTEND EACH MATCH ON TOP OF STACK THAT IS CONTIGUOUS WITH CURRENT CONTEXT INTERVAL
EXTEND EACH MATCH ATOP STACK SUCH THAT: WORD INTERVAL IS NEXT WORD IN PHRASE AND CONTIGUOUSLY EXTENDS MATCH (DISCARD ALL OTHER MATCHES)
564
IS CURRENT INTERVAL AN IGNORED ANNOTATION?
NO
YES
566
PUSH TO STACK: (CURRENT CONTEXT INTERVAL, EMPTY WITNESS SET, EMPTY MATCH SET)
5A
(TO FIG. 5A)

FIG. 5D

```
0.  /* Stack-Based Merge (SBM) Approach  */

1.  while (not(empty(L))) {
2.    i = remove-first(L);
3.    if (i ∈ LC) { /* i is context interval */
4.      if (not(empty(S))) &&
5.          not(descendant(i,top(S).interval)))
6.         output-and-clean(i);
7.    new-interval(i);
8.  } else { /* i is word or ignored markup */
9.    if (empty(S)) break;
10.  if (not(descendant(i, top(S).interval)))
11.     output-and-clean(i);
12.   /* i is descendant of top(S).interval */
13.   if (i ∈ LM) {
14.     extend-with-markup(i);
15.     if (i ∈ La_j)
16.       /* i is nested annotation */
17.       new-interval(i);
18.    } else if (i ∈ Lw pos)
19.      extend-with-word(i, pos)
20.   }
21. }
22. if (not(empty(S))) output-and-clean((0,0));
23.
24. output-and-clean(i) {
25.  repeat {
26.    c = pop(S);
27.    if (c.interval ∈ LC) /* context interval */
28.      output(c.interval,c.witnessSet);
29.    /* Propagate nested witnesses up stack */
30.    top(S).witnessSet =
31.      top(S).witnessSet U c.witnessSet;
32.  } until (empty(S) or
33.            descendant(i,top(S).interval));
34. }

35. new-interval(i) {
36.   push((i, {}, {}), S);
37. }
```

FIG. 6A

```
38.  discard-partial-match(m) {
39.   top(S).matchSet = top(S).matchSet - {m}
40.  }
41.  extend-with-markup(i) {
42.   for each m ∈ top(S).matchSet {
43.     if (i.start =
44.         last(m.partialWitness).end + 1)
45.       m.partialWitness = append(m.partialWitness, i);
46.     else
47.       discard-partial-match(m)
48.   }
49.  }
50.  extend-with-word(i, pos) {
51.   if (pos = 1) {
52.     top(S).matchSet = top(S).matchSet U ([ i ], 1);
53.   } else {
54.     for each m ∈ top(S).matchSet {
55.       if (m.matchPos + 1 = pos and i.start =
56.           last(m.partialWitness).end + 1) {
57.         m.partialWitness = append(m.partialWitness, i);
58.         m.matchPos++;
           /* Once matched complete phrase */
59.         if (m.matchPos = q) {
60.           /* Add to top witness set */
61.           top(S).witnessSet = top(S).witnessSet
                U { m.partialWitness }
62.           discard-partial-match(m)
63.         }
64.       } else
65.         discard-partial-match(m)
66.     }
67.   }
68.  }
```

FIG. 6B

```
0. /* Procedure for word-proximity matching */

1. extend-with-word(i, pos) {
2.  for each m ∈ top(S).matchSet {
3.    if (m.matchPos + 1 = pos and i.start =
4.         last(m.partialWitness).end + 1) {
5.       m.partialWitness =
           append(m.partialWitness, i);
6.       m.matchPos++;
         /* Once matched complete phrase */
7.       if (m.matchPos = q) {
8.         /* Add to top witness set */
9.         top(S).witnessSet = top(S).witnessSet
                U { m.partialWitness }
10.       discard-partial-match(m)
11.      }
12.   } else if (m.skipped + i.start -
           last(m.partialWitness).end - 1 <= k) {
13.      m.skipped += i.start -
           last(m.partialWitness).end - 1;
14.      m.partialWitness =
           append(m.partialWitness, i);
15.   } else {
16.      discard-partial-match(m)
17.   }
18.   if (pos = 1) {
19.      top(S).matchSet =
20.         top(S).matchSet U ([ i ], 1, 0);
21.   }
22.  }
23. }
```

FIG. 6C